Data Mining Exam

May 30, 2021

Kasper Rosenkrands

Aalborg University
Denmark

AALBORG UNIVERSITY
DENMARK

Clustering
Introduction

Clustering is a way to categorize data to impose structure.

A use case is recommender systems (Amazon, Spotify, Netflix), where a user is recommended items that bought/listened to/watched by other users with similar interests.

Clustering
K-Means Optimization Problem

Given $D = (x_1, \ldots, x_n)$ where $x_i \in \mathbb{R}^p$, $K \in \mathbb{N}$ and let $C_1, \ldots, C_K$ denote different groups of the x_i's.

The K-Means algorithm tries to solve

$$\min_{C_1,\ldots,C_K} \left\{ \sum_{k=1}^{K} W(C_k) \right\}, \tag{1}$$

where $W(C_k)$ denotes the **within cluster variation**, in other words the dissimilarity of the group.

The most common dissimilarity measure is the is the squared Euclidean distance

$$W(C_k) := \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{i,j} - x_{i',j})^2. \tag{2}$$

Clustering
K-Means Optimization Problem

If we by $\bar{x}_{k,j} = \frac{1}{|C_k|} \sum_{i \in C_k} x_{i,j}$ denote the mean value of the j'th dimension in cluster k, it can be shown that

$$\frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{i,j} - x_{i',j})^2 = 2 \sum_{i \in C_k} \sum_{j=1}^{p} (x_{i,j} - \bar{x}_{k,j})^2. \tag{3}$$

If we further note that $\bar{x}_{k,j} = \min_{\mu_k} \left\{ \sum_{i \in C_k} \sum_{j=1}^{p} (x_{i,j} - \mu_k)^2 \right\}$ this implies that the optimization problem in (1) can be rewritten as

$$\min_{C_1,\ldots,C_k,\mu_1,\ldots,\mu_k} \left\{ \sum_{k=1}^{K} \sum_{i \in C_k} \sum_{j=1}^{p} (x_{i,j} - \mu_k)^2 \right\}. \tag{4}$$

Clustering
K-Means Algorithm

The K-Means algorithm is now able to exploit the new formulation of the optimization problem and iteratively solve for $\{C_1, \ldots, C_k\}$ and $\{\mu_1, \ldots, \mu_k\}$.

This makes K-Means a greedy algorithm because, in each iteration it chooses optimal values for $\{C_1, \ldots, C_k\}$ and $\{\mu_1, \ldots, \mu_k\}$.

Convergence of the algorithm is therefore ensured, however we cannot guarantee it will find the global optimum.

Clustering
K-Means Algorithm

Algorithm 1: K-Means

```
1  Assign each obsevation to a cluster randomly
2  foreach Cluster do
3      Compute the centroid
4  foreach Observation do
5      Compute distance to all centroids
6      Assign to the closest
7  while Centroids have not changed since last iteration do
8      foreach Observation do
9          Compute distance to all centroids
10         Assign to the closest
11     foreach Cluster do
12         Compute the centroid
13 return Clusters
```

Clustering
An example of the K-Means algorithm

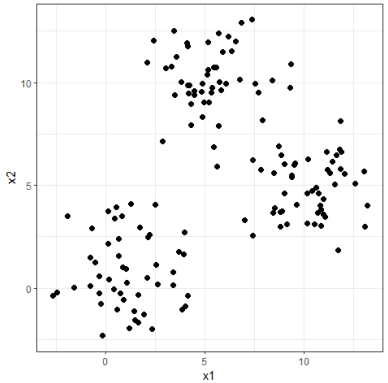


Figure: Iteration 01

Clustering
An example of the K-Means algorithm

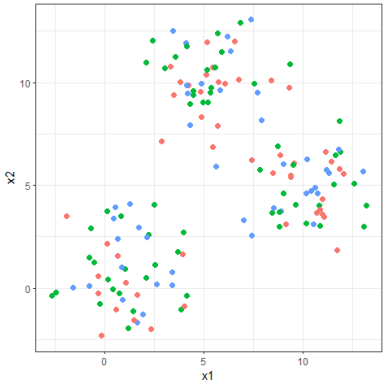


Figure: Iteration 02

Clustering
An example of the K-Means algorithm

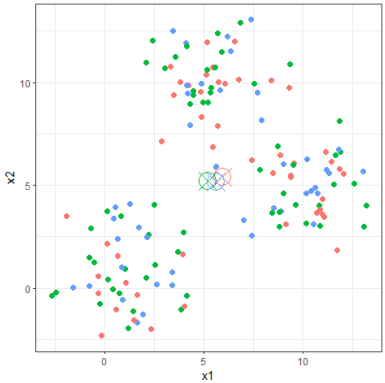


Figure: Iteration 03

Clustering
An example of the K-Means algorithm

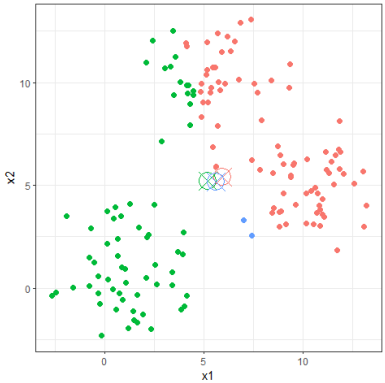


Figure: Iteration 04

Clustering
An example of the K-Means algorithm

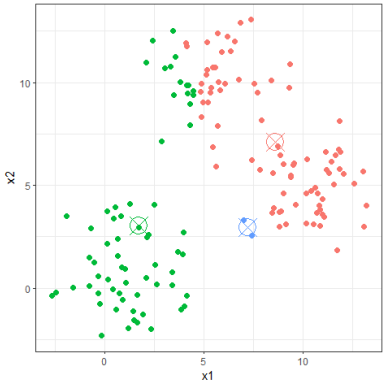


Figure: Iteration 05

Clustering
An example of the K-Means algorithm

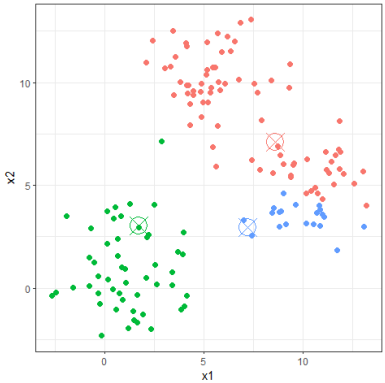


Figure: Iteration 06

Clustering
An example of the K-Means algorithm

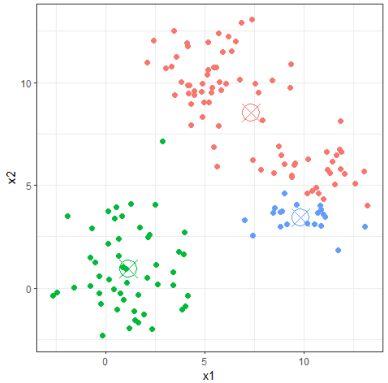


Figure: Iteration 07

Clustering
An example of the K-Means algorithm

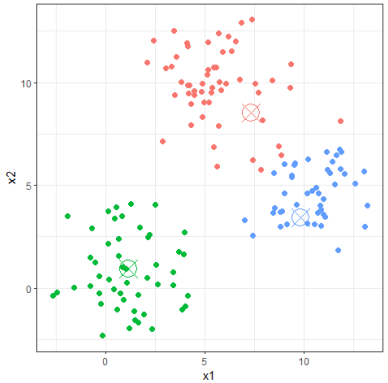


Figure: Iteration 08

Clustering
An example of the K-Means algorithm

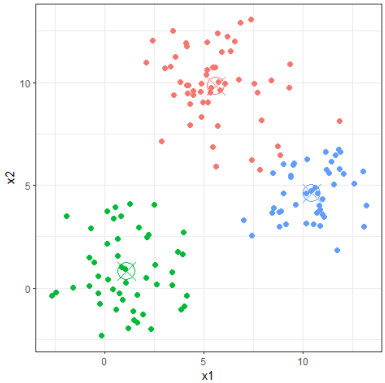


Figure: Iteration 09

Clustering
An example of the K-Means algorithm

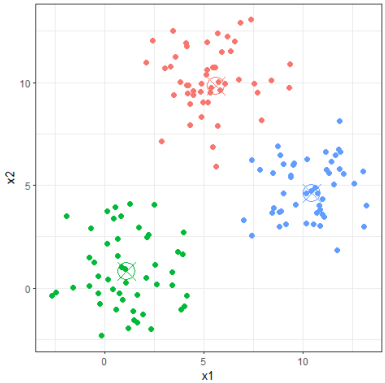


Figure: Iteration 10

Clustering
An example of the K-Means algorithm

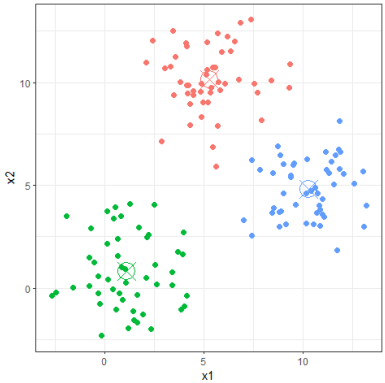


Figure: Iteration 11

Clustering
An example of the K-Means algorithm

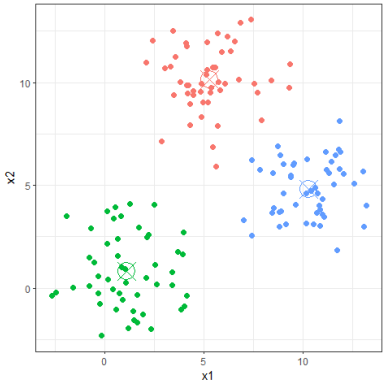


Figure: Iteration 12

Clustering
An example of the K-Means algorithm

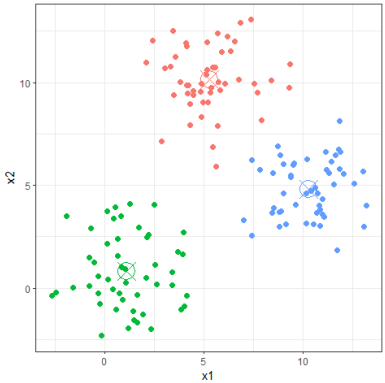


Figure: Iteration 13

Clustering
Number of clusters

Prior to running K-Means we need to determine the number of clusters. For some use cases this number may be predetermined by external factors. If the number of clusters is not predetermined, the data itself can indicate the optimal number.

The **total sum of squares** (total variance in the data) and the **within cluster sum of squares** are given by

$$SS_{total} = \sum_{i=1}^{n}(x_i - \bar{x})^2, \quad \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad SS_{W_k} = \sum_{i \in C_k}(x_i - \bar{x}_k)^2, \quad \bar{x}_k = \frac{1}{n_k}\sum_{i \in C_k} x_i.$$

Then the **total within sum of squares** becomes $SS_W = \sum_{k=1}^{K} SS_{W_k}$. Using this we can calculate the **percentage of variance explained** as $PVE = SS_W / SS_{total}$.

Clustering
Number of clusters

We can then use what is called the **elbow method** to determine number of clusters.

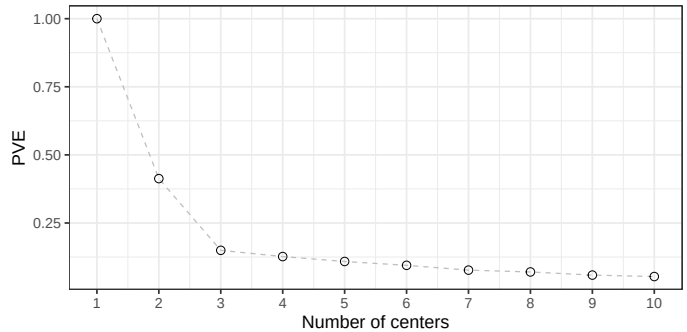
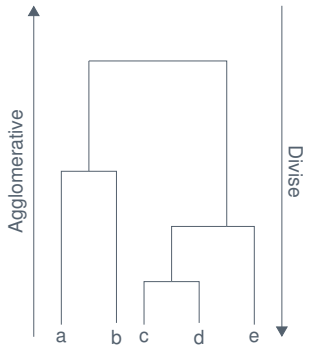


Figure: Scree plot showing the number of clusters against percentage of variance explained.

Clustering
Hierarchical Clustering

Hierarchical clustering is a method which seeks to build a hierarchy of clusters.
In general there are two approaches to obtain this hierarchy, starting from the bottom or from the top (so to speak):

1. **Agglomerative**: This is the bottom up approach, it starts by having each observation in its own cluster and then it merges the most similar together.
2. **Divise**: This is the top down approach, it start by putting all observations in one cluster and then splits recursively to obtain more and more homogenous clusters.



Clustering
Linkages

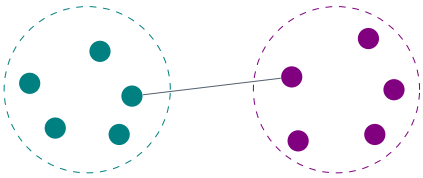
In order to quantify the similarity between observations we need to establish a dissimilarity measure. However there are many ways to define dissimilarity, or linkage, between groups of observations.

1. **Single**: The smallest pairwise dissimilarity.
2. **Complete**: The largest pairwise dissimilarity.
3. **Average**: The average dissimilarity between observations.
4. **Centroid**: The dissimilarity between centroids of clusters.[1]

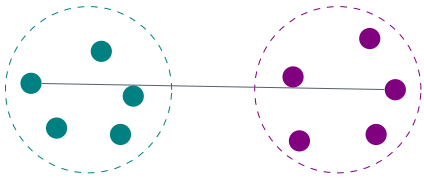
Another consideration is what distance measure to use, typically the Euclidean distance is used, but for other use cases one might opt for the Manhattan distance etc.

[1] Squared distance can cause inversions as we “create” new data points.

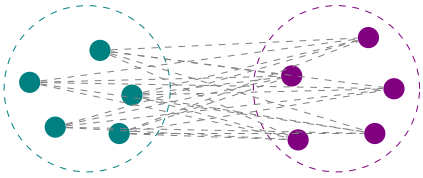
Clustering
Linkages



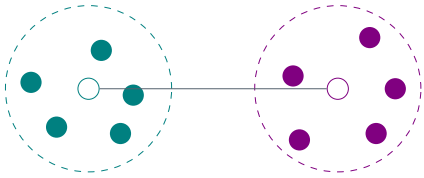
(a) Single.



(b) Complete.



(c) Average.



(d) Centroid.

Figure: The 4 different types of linkages presented on the slide before.

Clustering
Linkages

Shrinkage
Variable Selection and Regularization

Because it is often cheaper to obtain multiple observations from a few samples than to obtain more samples, thus increasing the number of explanatory variables in a regression model, a linear regression model would be prone to increasing variance.

When extending linear regression to multiple exaplanatory variables the main objectives is:

- **Model Interpretability**: Models with fewer variables are often more easy to interpret results from and are therefore better to use for decision making.
- **Predition Accuracy**: If by introducing some bias we are able to dramatically improve prediction accuracy, this would be worth considering (bias-variance tradeoff).

Shrinkage
Variable Selection and Regularization

There are multiple tools we can use in this pursuit

- **Subset Selection**: Works by fitting lots of models with different combinations of predictors. Then we can find out which variables are most related to the response and we can select these.
- **Dimensionality Reduction**: Works by projecting explanatory variables into a smaller dimensional space and use these projections as predictors.
- **Shrinkage Methods**: Works by fitting a model using all predictors while shrinking coefficients towards zero, to reduce variance. Some shrinkage methods can also perform variable selection by shrinking coefficients to exactly zero.

Shrinkage
Variable Selection and Regularization

This presentation will focus on shrinkage methods. The two main shrinkage methods are

- **Ridge Regression**
- **Lasso**

They both penalize the “size” of the estimated parameters, however they difer in the way they quantify the “size”. Ridge regression penalize the ℓ_2 norm while Lasso penalize the ℓ_1 norm.

Therefore we can also refer to the two methods as ℓ_1- and ℓ_2-regularizations.

Shrinkage
Ridge Regression

The expression that ridge regression looks to minimize is the following

$$\underbrace{\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2}_{RSS} + \underbrace{\lambda\sum_{j=1}^{p}\beta_j^2}_{Penalty}, \tag{5}$$

where λ is a parameter that needs to be determined seperately, this is usually done by cross validation.

When $\lambda = 0$ there is no penalty and we are just doing linear regression (OLS). If $\lambda \to \infty$ then β_0 will approach the mean and $\beta_j \to 0$.

Shrinkage

Variable Selection and Regularization

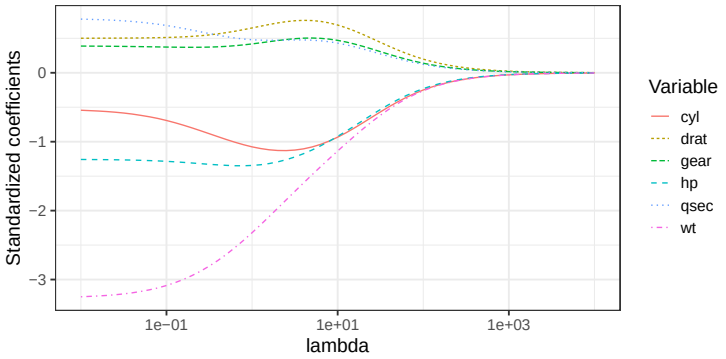


Figure: The effect of λ on coefficents in a ridge regression model for the `mtcars` dataset.

Shrinkage
Variable Selection and Regularization

A benefit of ridge regression is that it can be solved analytically, as the minimization problem is

$$RSS(\lambda) = (\mathbb{Y} - \mathbb{X}\beta)^\top(\mathbb{Y} - \mathbb{X}\beta) - \lambda\beta^\top\beta. \tag{6}$$

If we the above function be the lagrange function, differentiate w.r.t. β and set this equal to zero we find that

$$\hat{\beta}^R = (\mathbb{X}^\top\mathbb{X} + \lambda I)^{-1}\mathbb{X}^\top\mathbb{Y}. \tag{7}$$

It is easy to compute as it is just a closed form expression and we can again observe that for $\lambda = 0$ we get the OLS estimate.

Shrinkage
Lasso

The expression that lasso looks to minimize is the following

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j|,$$

where λ, again, is a parameter that needs to be determined seperately.

As opposed to ridge regression, lasso can force coefficients to exactly zero and thereby perform variable selection.

Shrinkage
Lasso

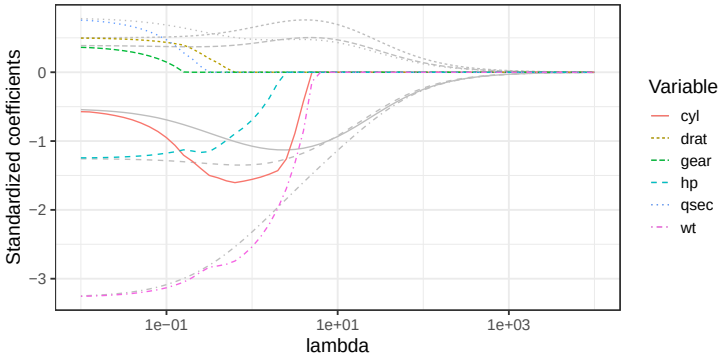


Figure: The effect of λ on coefficents in a lasso regression model for the `mtcars` dataset. Ridge coefficients are included as gray lines for reference.

Shrinkage
Lasso

Unlike ridge regression, there does not exists an analytical solution for the lasso estimates. This is because the absolute value is **not** differentiable.

Therefore we must resort to numerical optimization.

For this we can use a method called **coordinate descent**, that works in the following way:

Instead of trying to solve for all parameters at once we will look for a solution one dimension at a time. When we are done with one dimension we go to the next and then iterate through until we converge.

Shrinkage
Lasso

For the following computations we will assume that the data have been demeaned so we can disregard β_0.

We then start the coordinate descent by the following minimization problem (orange term is just to simplify computation, N is the number of observations and p is the number of variables)

$$\min_{\beta_j} \left\{ \frac{1}{2N} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} x_j \beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}. \tag{8}$$

Let us define a function L as

$$L = \frac{1}{2N} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} x_j \beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \tag{9}$$

Shrinkage
Lasso

If we then differentiate w.r.t. β_k we get

$$\frac{\partial L}{\partial \beta_k} = N^{-1} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} x_j \beta_j \right) (-x_k) + \lambda \underbrace{\partial |\beta_k|}_{\text{Subgradient}} \tag{10}$$

$$= N^{-1} \sum_{i=1}^{N} \left(y_i - \sum_{j \neq k}^{p} x_j \beta_j - x_k \beta_k \right) (-x_k) + \lambda \partial |\beta_k| \tag{11}$$

$$= N^{-1} \sum_{i=1}^{N} \left(y_i - \sum_{j \neq k}^{p} x_j \beta_j \right) (-x_k) + N^{-1} \beta_k \sum_{i=1}^{N} x_k^2 + \lambda \partial |\beta_k| \tag{12}$$

We can define the purple term as $r_k = y_i - \sum_{j \neq k}^{p} x_j \beta_j$ as this is the residual from the model w.o. the k'th regressor.

Shrinkage
Lasso

We can then write

$$= -N^{-1} \sum_{i=1}^{N} (r_k x_k) + N^{-1} \beta_k \sum_{i=1}^{N} x_k^2 + \lambda \partial |\beta_k| \tag{13}$$

$$= -N^{-1} r_k^\top x_k + N^{-1} x_k^\top x_k \beta_k + \lambda \partial |\beta_k|, \tag{14}$$

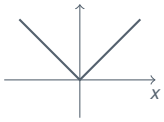
So now we need to figure out what to do with the subgradient.

Shrinkage
Lasso

As there are lots of lines tangent to the absolute value in zero, we can summarize the subgradient as

$$\lambda \partial |\beta_k| = \begin{cases} -\lambda & \text{if } \beta_k < 0 \\ [-\lambda, \lambda] & \text{if } \beta_k = 0 \\ \lambda & \text{if } \beta_k > 0 \end{cases}, \tag{15}$$

for $\beta_k = 0$ the gradient can take all values between -1 and 1.



(a) $f(x) = |x|$.

(b) $\partial f(x)$.

Figure: The absolute value and its gradient.

Shrinkage
Lasso

This means that we can rewrite the derivative as the following

$$\frac{\partial L}{\partial \beta_k} = \begin{cases} -N^{-1} r_k^\top x_k + N^{-1} x_k^\top x_k \beta_k - \lambda & \text{if } \beta_k < 0 \\ [-N^{-1} r_k^\top x_k - \lambda, -N^{-1} r_k^\top x_k - \lambda] & \text{if } \beta_k = 0 \\ -N^{-1} r_k^\top x_k + N^{-1} x_k^\top x_k \beta_k + \lambda & \text{if } \beta_k > 0 \end{cases}. \tag{16}$$

Setting this equal to zero and solving for β_k in the 3 cases gives us the estimates, we take the case $\beta_k < 0$ as an example

$$\frac{\partial L}{\partial \beta_k} = 0 \Leftrightarrow -N^{-1} r_k^\top x_k + N^{-1} x_k^\top x_k \beta_k - \lambda = 0 \tag{17}$$

$$N^{-1} x_k^\top x_k \beta_k = N^{-1} r_k^\top x_k + \lambda \tag{18}$$

$$\beta_k = \left(N^{-1} x_k^\top x_k\right)^{-1} \left(N^{-1} r_k^\top x_k + \lambda\right). \tag{19}$$

Shrinkage
Lasso

Note that

$$\beta_k < 0 \quad \Leftrightarrow \quad N^{-1} r_k^\top x_k + \lambda < 0 \quad \Leftrightarrow \quad N^{-1} r_k^\top x_k < -\lambda. \tag{20}$$

Ultimately we find that, these are the values to update parameters by

$$\hat{\beta}_k = \begin{cases} \left(N^{-1} x_k^\top x_k\right)^{-1} \left(N^{-1} r_k^\top x_k + \lambda\right) & \text{if } N^{-1} r_k^\top x_k < -\lambda \\ 0 & \text{if } -\lambda \leq N^{-1} r_k^\top x_k \leq \lambda \\ \left(N^{-1} x_k^\top x_k\right)^{-1} \left(N^{-1} r_k^\top x_k - \lambda\right) & \text{if } N^{-1} r_k^\top x_k > \lambda \end{cases}, \tag{21}$$

and we remember that r_k is the residual from the model w.o. the k'th regressor.

Shrinkage
Elastic Net

The method is a compromise between Ridge Regression and Lasso as it minimizes

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\left(\alpha|\beta_j| + (1-\alpha)\beta_j^2\right),$$

note that here we have the additional parameter α that determines the weighting of the two norms.

Shrinkage
Elastic Net

- The figure illustrates the feasible region, in the case of two variables, on the RSS surface.
- Most likely the OLS estimate will be somwhere outside the shapes plotted.
- A lasso estimate is more likely to be at the edges.
- Elastic net keeps the pointy edges but rounds the sides (depending on α).
- Elastic net performs variable selection like the lasso, but shrinks correlated regressors like ridge.

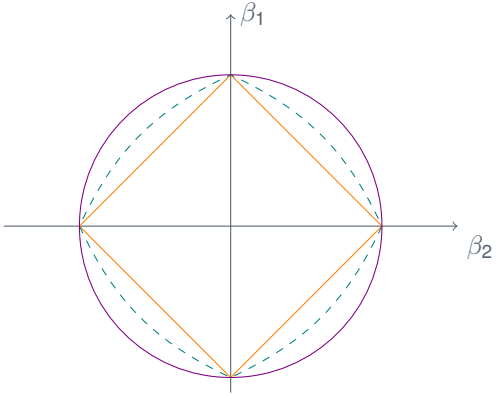


Figure: The feasible region for ridge, lasso and elastic net ($\alpha = .5$).

Shrinkage
Scaling of variables

For both shrinkage methods it is important to scale the variables before fitting the model.

Contrary to linear regression where the scaling of variables are absorbed in the coefficients, this will affect the penalty term for both shrinkage methods.

For example if a variable takes very large values it will get a higher penalty than a variable that takes small values, just because of the unit of measurement.

Classification
Overview

Add something about logit and naive bayes, very brief

Maybe include example with an ROC curve

Classification
Introduction

If the response variable is categorical (qualitative), i.e. it is of the form $y \in \{1, \ldots, L\}$. Then a linear model of the form

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p,$$

is generally not a good approach to take as it could predict invalid values and for certain types of categorical data there might not be a clear ordering.

Therefore when dealing with categorical variable the aim of the model is to predict the probability that an observation belongs to a certain category, rather than the category itself.

Classification
Logistic Regression

To begin with let us assume the simple case that $y \in \{0, 1\}$ and that we would like to model

$$p(Y = 1|X_i) = p(X_i) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p = X_i\mathbb{B}. \tag{22}$$

Given that we know probabilities lie between 0 and 1, we would like to transform the linear function such that the output is between 0 and 1.

The most popular, although there are several, would be the logistic function

$$p(X_i) = \frac{e^{X_i\mathbb{B}}}{1 + e^{X_i\mathbb{B}}}. \tag{23}$$

For $X_i\beta \to \infty$ then $p(X_i) \to 1$ and for $X_i\beta \to -\infty$ then $p(X_i) \to 0$.

Classification
Linear Discriminant Analysis (LDA)

Suppose we have a dataset with a response variable $y \in \{0, \ldots, L\}$, explanatory variables $x_1, \ldots, x_p$ and that we would like to model

$$P(y = k|X) = \underbrace{\frac{P(y = k)P(X|y = k)}{P(X)}}_{\text{Bayes Theorem}} = \frac{\pi_k f_k(x)}{\sum_{i=1}^{K} f_i(x)}, \tag{24}$$

where $\pi_k = P(y = k)$ and $f_k(x) = P(X|y = k)$.

If we use the proportion of observations in the dataset that belong to class k as an estimate for π_k, then we just need to model $f_k(x) = P(X|y = k)$.

In other words we need to make some assumption on the distribution of $f_k(x)$.

Classification
Linear Discriminant Analysis (LDA)

The assumption made in LDA is that each $f_k(x)$ come from a multivariate normal distribution, i.e.

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_k)^\top \Sigma_k^{-1}(x-\mu_k)\right\}. \tag{25}$$

Another assumption made in LDA is that $\Sigma_k = \Sigma \; \forall k \in \{1, \ldots, K\}$, in other words all classes will have the same variance-covariance matrix.

Classification
Linear Discriminant Analysis (LDA)

When we classify a new observation, X_0, we simply find the category with the highest probability, $P(y = k|X_0)$ for $k = 1, \ldots, K$.

In other words we want to find the k such that $P(y = k|X)$ is maximized. Since the logarithm is an increasing function we know that the k which maximizes $P(y = k|X)$ also maximizes the following

$$\log(P(y = k|X)) = \log(\pi_k) + \log(f_k(x)) - \underbrace{\log\left(\sum_{i=1}^{K} \pi_i f_i(x)\right)}_{\text{Does not depend on } k}, \tag{26}$$

The last term is identical across categories and we drop this term for the maximization problem.

Classification
Linear Discriminant Analysis (LDA)

So now we have the following

$$\log(\pi_k) + \log(f_k(x)), \tag{27}$$

but let us take a look at the second term. By the normality assumption we have that

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu_k)\right\}. \tag{28}$$

Thus we can write (27) as

$$\log(\pi_k) + \underbrace{\log((2\pi)^{p/2}|\Sigma|^{1/2})}_{\text{Does not depend on } k} - \frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu_k), \tag{29}$$

and subsequently drop another term that is identical across categories.

Classification
Linear Discriminant Analysis (LDA)

So now we have the following

$$\log(\pi_k) - \frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu_k), \tag{30}$$

again taking a look at the second term

$$(x-\mu_k)^\top \Sigma^{-1}(x-\mu_k) = \underbrace{x^\top \Sigma^{-1} x}_{\text{Does not depend on } k} - x^\top \Sigma^{-1}\mu_k - \mu_k^\top \Sigma^{-1} x + \mu_k^\top \Sigma^{-1}\mu_k, \tag{31}$$

and furthermore we have that $x^\top \Sigma^{-1}\mu_k = \mu_k^\top \Sigma^{-1} x$ because both are scalars and one is just the transpose of the other.

Classification
Linear Discriminant Analysis (LDA)

So we end up with the following expression, which is called the **linear discriminant function**

$$\delta_k(x) = \log(\pi_k) + x^\top \Sigma^- 1 \mu_k - \frac{1}{2}\mu_k^\top \Sigma^{-1} \mu_k, \tag{32}$$

this is what we will use to classify observations. Note the expression is linear in x and therefore we call it linear discriminant analysis.

Classification
Linear Discriminant Analysis (LDA)

In practice the parameters are estimated by

$$\hat{\pi}_k = \frac{n_k}{n}, \quad \hat{\mu}_k = \frac{1}{n_k} \sum_{y_i = k} x_i, \quad \hat{\Sigma} \frac{1}{n - K} \sum_{k=1}^{K} \sum_{y_i = k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^\top, \tag{33}$$

i.e. the proportion of observations in class k, the mean for class k and the mean of the variances respectively.

Classification
Quadratic Discriminant Analysis (QDA)

Imagine now that we relax the LDA assumption that $\Sigma_k = \Sigma \; \forall k \in \{1, \ldots, K\}$ and allow classes to have their own variance-covariance matrix.

This implies that some of the simplification we performed before no longer holds and the discriminant function will now take the form

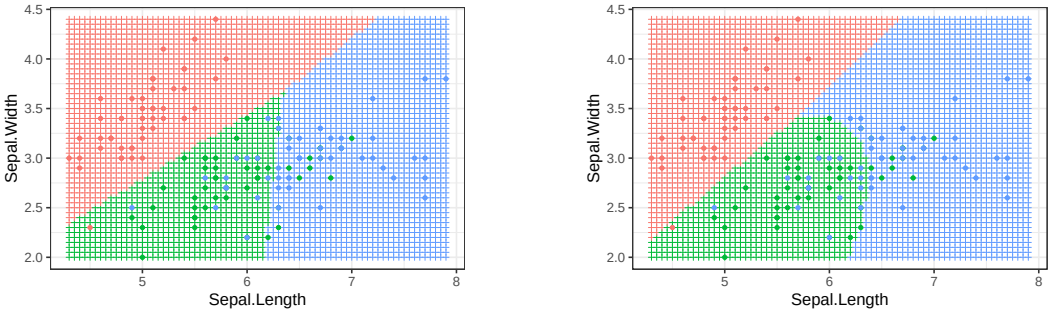
$$\delta_k(x) = -\frac{1}{2} x^\top \Sigma_k x + \log(\pi_k) + x^\top \Sigma_k{}^{-1} \mu_k - \frac{1}{2} \mu_k^\top \Sigma_k{}^{-1} \mu_k - \frac{1}{2} \log |\Sigma_k|, \tag{34}$$

note that first term is quadratic in x.

Another difference in the estimation of the covariance matrices, because we relaxed the assumption we must estimate a variance-covariance matrix for each class

$$\hat{\Sigma}_k = \frac{1}{n_k - 1} \sum_{y_i = k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^\top. \tag{35}$$

Classification
Example using the Iris dataset



(a) LDA classification regions.

(b) QDA classification regions.

Figure: Classification regions for LDA and QDA performed on the *Iris* dataset, considering only the variables *sepal width* and *sepal length*. Note that the LDA boundaries are linear while QDA include a non-linear boundary.

Classification
Example using Auto dataset

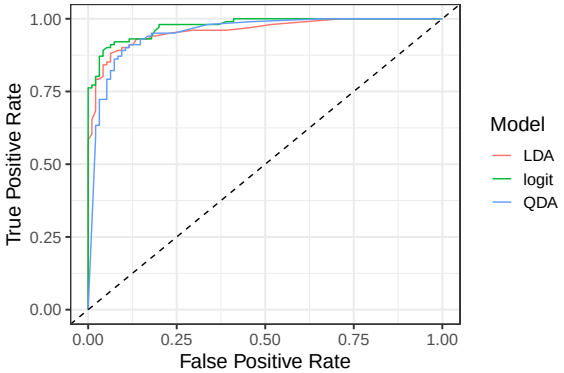


Figure: ROC curves for 3 models on the *auto* dataset from the `ISLR` package. We are predicting whether a car have mpg above the median using displacement, weight, year and horsepower.

Classification
Naive Bayes

The (Naive) Bayes classifier is an example of a parametric classifier (meaning that the number of parameters is fixed), it builds on Bayes theorem

$$P(Y = i|X) = \frac{P(Y = i)P(X|Y = i)}{P(X)}. \tag{36}$$

New data is then assigned to the class with the highest probability given the explanatory variables

$$\hat{Y} = \ arg\ max_i P(Y = i)P(X|Y = i). \tag{37}$$

This means that there are two things we need to model, the first is $P(Y = i)$, but this is simply modelled by the proportions of the dataset.

Classification
Naive Bayes

The second one is $P(X|Y=i)$, note that X is a p-dimensional vector, making the distribution multivariate.

However as estimating multivariate distribution requires lots of computation and suffers from the curse of dimensionality, an assumption in Naive Bayes is that X is independent between classes

$$P(X|Y_i) = P(X_1|Y)P(X_2|Y)\cdots P(X_p|Y). \tag{38}$$

With this assumption we can rewrite

$$P(X|Y=i) = \prod_{j=1}^{p} P(X_j|Y=i), \tag{39}$$

which enables us to estimate all marginal (univariate) distributions one by one.

Classification
Naive Bayes

Typically the we assume that the ditributions are normal, but we can also use others and even seperate distributions for each explanatory variable.

In the case of normally distributed variables we only need to estimate the mean and variance as $X_j|Y = i \sim N(\mu_{j,i}, \sigma^2_{j,i})$, where we use the usual estimates

$$\mu_{j,i} = \frac{1}{N_i} \sum_{x_k|y_k=i} x_{j,i}, \quad \sigma^2_{j,i} = \frac{1}{N_i} \sum_{x_k|y_k=i} (x_{j,i} - \mu_{j,i})^2. \tag{40}$$

Despite the assumption of independence the method tends to work well in practice, although if the assumption is clearly not true we will get biased estimator.

Trees
Overview

Trees
Introduction

Tree-based methods are supervised learning methods which can be used for both regression and classification.

They work by segmenting the space into a number of simple regions and then, typically, use the mean of the region to make predictions.

Because the segmentation of the space can be summarized in a tree, we call these methods tree-based.

Trees

Classification and Regression Trees (CART)

The purpose of the CART algorithm is to decide on the split points. This is done in a greedy way one split at a time, as the task of checking all possible combinations of splits would be computationally infeasible.

Trees
Classification and Regression Trees (CART)

Consider the following model (non-binary)

$$y = 1 + 2x_1 + 3x_2 + \varepsilon, \tag{41}$$

where $x_1, x_2 \sim N(0,1)$ and $\varepsilon \sim N(0, 1/2)$. We can then introduce the variables $\tilde{x}_1, \tilde{x}_2$ given by

$$\tilde{x}_1 = \begin{cases} 1 & \text{for } x_1 > 0 \\ 0 & \text{for } x_1 \leq 0 \end{cases}, \quad \tilde{x}_2 = \begin{cases} 1 & \text{for } x_2 > 0 \\ 0 & \text{for } x_2 \leq 0 \end{cases}, \tag{42}$$

and proprose a separate model (binary)

$$\tilde{y} = 1 + 2\tilde{x}_1 + 3\tilde{x}_2 + \varepsilon. \tag{43}$$

We can then compare how the CART algorithm compares to linear regression on both the binary and non-binary model.

Trees

Classification and Regression Trees (CART)

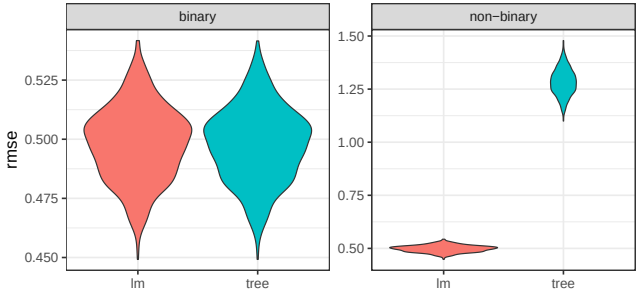


Figure: Comparing linear regression (`lm` from `base R`) and CART (`tree` from the `tree` package) on 1000 repetions of 500 observations. We see that when the data is generated from a step function its a tie between `tree` and `lm`, but when the data is linear the CART understandably struggles compared to linear regression.

Trees
Bagging

Trees
Random Forest

Trees
Boosting

Support Vector Machines
Overview

Neural Networks
Overview

Neural Networks
Perceptron

Backpropagation: For a given loss function L we look for $\frac{\partial L}{\partial w_i}$. We start with initial values for the weights, which we shall denote w_{old}. Then we update the weights by $w_{new} = w_{old} - \eta \frac{\partial L}{\partial w}$. One iteration is called an **epoch**.